INF404 - Travaux pratiques - Séance 4

Expressions arithmétiques "générales" (partie 1)

Avant de commencer cette séance :

1. Créez un répertoire ***TP3*** dans votre répertoire ***INF404***
2. Placez-vous dans ***INF404/TP3*** et recopiez les fichiers utilisés pendant le TP2 : `cp ../TP2/* .`

Objectifs

L'objectif de cette séance est de programmer un analyseur syntaxique pour des expressions arithmétiques "générales" (avec les opérateurs $+$, $-$ et $*$, ainsi que les parenthèses). On rappelle la grammaire vue en cours pour définir la syntaxe de ces expressions :

$$
\begin{aligned}
exp &\rightarrow eag\ \texttt{FIN\_DE\_SEQUENCE}\\
eag &\rightarrow seq\_terme\\
seq\_terme &\rightarrow terme\ suite\_seq\_terme\\
suite\_seq\_terme &\rightarrow op1\ terme\ suite\_seq\_terme\\
suite\_seq\_terme &\rightarrow \varepsilon\\
terme &\rightarrow seq\_facteur\\
seq\_facteur &\rightarrow facteur\ suite\_seq\_facteur\\
suite\_seq\_facteur &\rightarrow op2\ facteur\ suite\_seq\_facteur\\
suite\_seq\_facteur &\rightarrow \varepsilon\\
facteur &\rightarrow ENTIER\\
facteur &\rightarrow \texttt{PARO}\ eag\ \texttt{PARF}\\
op1 &\rightarrow \texttt{PLUS}\\
op1 &\rightarrow \texttt{MOINS}\\
op2 &\rightarrow \texttt{MUL}
\end{aligned}
$$

Exercice 1 - analyse syntaxique d'une EAG

Modifier la fonction ***analyse_syntaxique*** du TP2 afin de reconnaitre une EAG (au lieu d'une EAEP) :

```
void analyser (char *nom_fichier) ;
   -- e.i : indifferent
   -- e.f : une EAG a ete lue dans le fichier de nom nom_fichier
   -- un message est affiche pour indiquer si cette expression contenait
   -- ou non une erreur de syntaxe
```

Cette fonction sera écrite à l'aide de fonctions récursives `Rec_exp()`, `Rec_eag()`, `Rec_seq_terme()`, etc. N'hésitez pas à vous aider des transparents du cours !

Modifiez alors le programme principal du TP2 pour tester votre analyse syntaxique sur divers exemples d'expression (corrects et incorrects). **Attention**, on ne demande pas de calculer mla **valeur** de l'expression, juste de vérifier sa syntaxe ...

Exercice 2 - ajouter la division

En vous inspirant des règles de la grammaires concernant les opérateurs d'addition et de soustraction, complétez cette grammaire pour ajouter un opérateur de division (il faudra le cas échéant compléter aussi l'analyse lexicale).

Modifiez alors la procédure `analyser` pour prendre en compte cette nouvelle grammaire.

Exercice 3 [si vous avez du temps, pour préparer la suite]

Téléchargez depuis Moodle les fichiers suivants dans le répertoire courant :
https://im2ag-moodle.univ-grenoble-alpes.fr/mod/resource/view.php?id=23968

- **type_ast.h**, spécification d'un arbre abstrait ;
- **ast_construction.h** et **ast_construction.c**, spécification et corps des primitives de construction d'un arbre abstrait ;
- **ast_parcours.h**, **ast_parcours.c**, spécification et corps des primitives de parcours d'un arbre abstrait ;
- **essai_ast.c**, programme principal (de test).

Examinez ces fichiers ...

Complétez le corps de la fonction `evaluation` du fichier **ast_parcours.c** contenant le corps des fonctions définies dans **ast_parcours.h**.

A l'aide du `Makefile` fourni, compilez le programme principal **essai_ast.c**, et exécutez-le. Vérifiez qu'il affiche bien l'expression arithmétique correspondant à l'arbre défini dans la variable `expression` ainsi que sa `valeur correcte`.

Testez ensuite votre programme sur les expressions suivantes (en modifiant **essai_ast.c** pour construire les arbre abstraits correspondants) :

- `2 + 3 * 5 - 2`
- `(2 + 3) * (5 - 2)`
- `2 + 3 / (5 - 2)`
- `3 / (5 - 2 - 3)`

Compte-rendu

Le compte-rendu de cette séance sera groupé avec celui des prochaines séances de TP, lorsque le TP3 sera terminé ...